A Cost Model for Data Stream Processing on Modern Hardware

Constantin Pohl, Philipp Götze, Kai-Uwe Sattler

Motivation and Introduction

- Main goals on Data Stream Processing Queries:
  High throughput & **low latency**

- Responsibility: Optimizer, based on cost model

- New challenges and opportunities in modern HW (like Manycore CPUs or NVRAM technology)

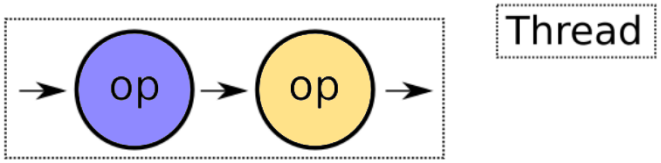
- Cost model with most important HW factors and Stream Processing Engine (SPE) characteristics

General idea (Optimizer)

- Hardware with generally valid properties, e.g.
  - **CPU**: clock frequency, available cores/threads,…
  - **Memory**: size, latency,…
- Hardware calibration tool for measuring relevant factors on different systems in advance
- Known measurements & memory accessing costs for SPE operators (joins, aggregations, etc.)
- Result: Cost model for parametrization avoiding worst-case performance

Query Costs

- Costs expressed as „necessary time to process a certain amount of tuples“
- Transformation into „avg. latency per processed tuple“
- Single thread for query

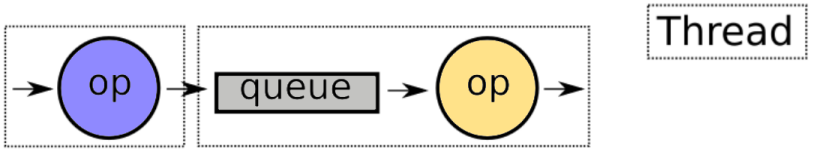


Costs: $c_q = \sum_{i=1}^{n} \left(c_{op(i)} \cdot tp_{\text{in}} \right)$

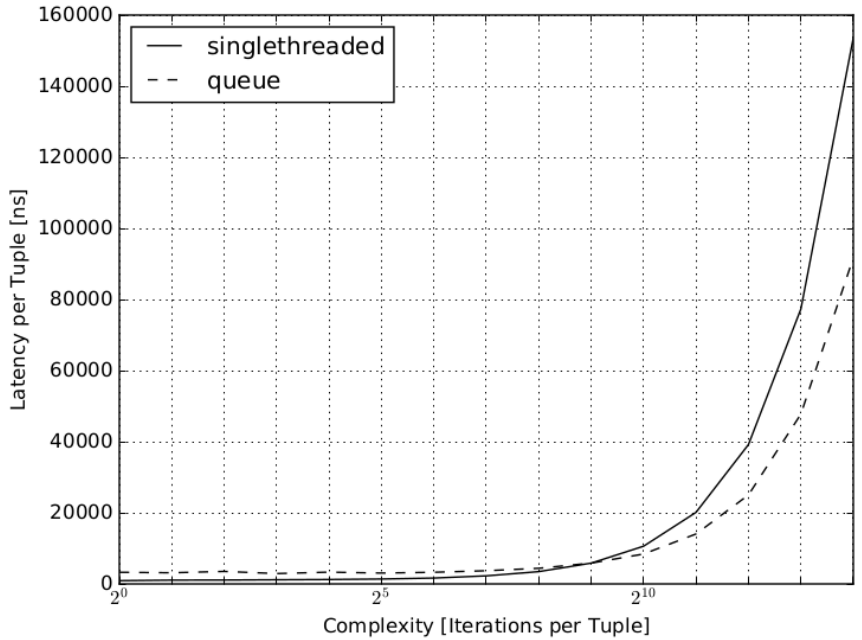
1...n: number of query operators

tp_{in}: number of processed tuples

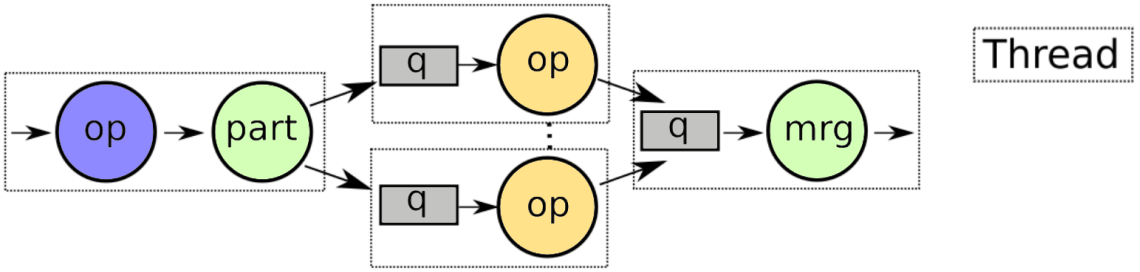
Inter-Operator Parallelism



Costs: $$c_q = max\left(\sum_{i=1}^{k}\left(c_{op(i)} \cdot tp_{\text{in}}\right), \sum_{i=k+1}^{n}\left(c_{op(i)} \cdot tp_{\text{in}}\right)\right) + c_{queue} \cdot tp_{\text{in}}$$

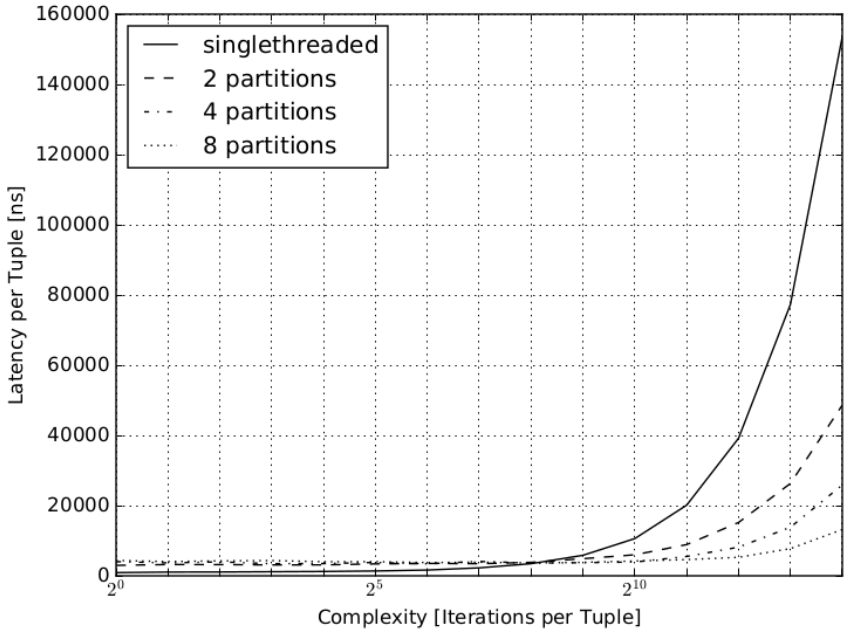


Intra-Operator Parallelism



Costs:

$$c_q = max\left(\sum_{i=1}^{k-1}\left(c_{op(i)} \cdot tp_{\text{in}}\right) + c_p \cdot tp_{\text{in}}, \left(c_{queue} + c_{op(k)}\right) \cdot tp_{\text{in}}, \left(c_{queue} + c_m\right) \cdot tp_{\text{in}} + \sum_{i=k+1}^{n}\left(c_{op(i)} \cdot tp_{\text{in}}\right)\right)$$



Operator Costs

- Dependence on Algorithms/Implementation
- Different for used Stream Processing Engine, here: *PipeFabric*[1]
- Example: **Projection** (reducing number of tuple attributes)

$$c_{op(Proj)} = L1_{latency} + mem_{latency} + \frac{c_{CPU}}{f_{clock}}$$

Accessing tuple in L1 Cache

Writing Projection result in main memory

Clock frequency of CPU (Instructions per time unit)

CPU instructions (function calls, publish/subscribe tuple from predecessing/following operators, ...)

[1] https://github.com/dbis-ilm/pipefabric

Hardware Facts (Manycore CPU)

Intel Xeon Phi Knights Landing (**KNL**, 2016)

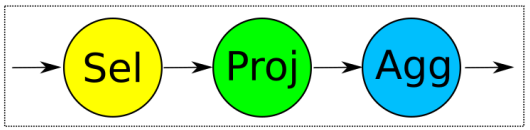
Some features:

- <72 cores à 4 threads, <1.5 GHz
- <384GB DDR4, 8-16GB MCDRAM (400+ GB/s)

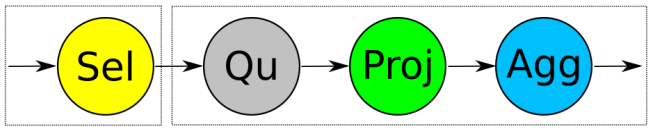
Test Queries

- Tuples: *Integer, String, Double* attributes
- Query 1:
  - Selection (20% selectivity)
  - Projection (Integer, Double)
  - Aggregation (Double)
- Threads:

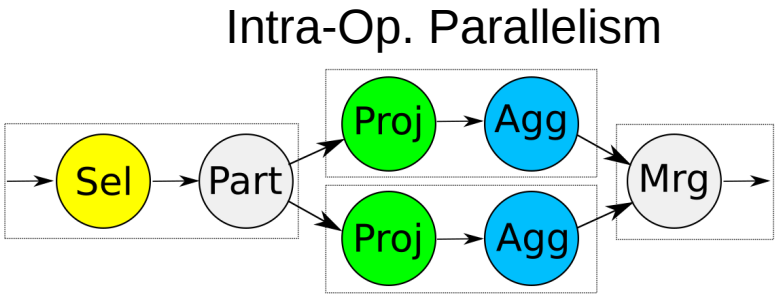
Single-Threaded



Inter-Op. Parallelism



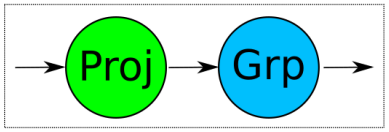
Intra-Op. Parallelism



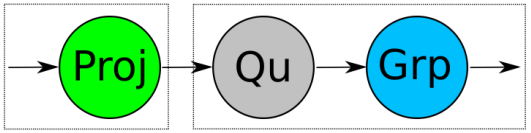
Test Queries

- Tuples: *Integer, String, Double* attributes
- Query 2:
  - Projection (Integer, Double)
  - Grouping (Double)
- Threads:

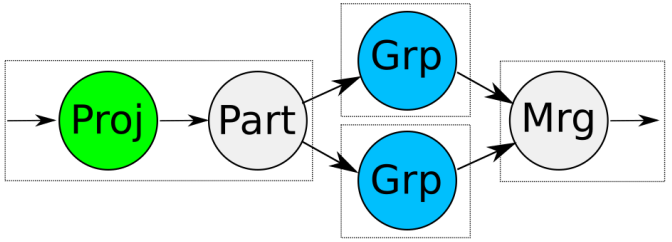
Single-Threaded



Inter-Op. Parallelism



Intra-Op. Parallelism



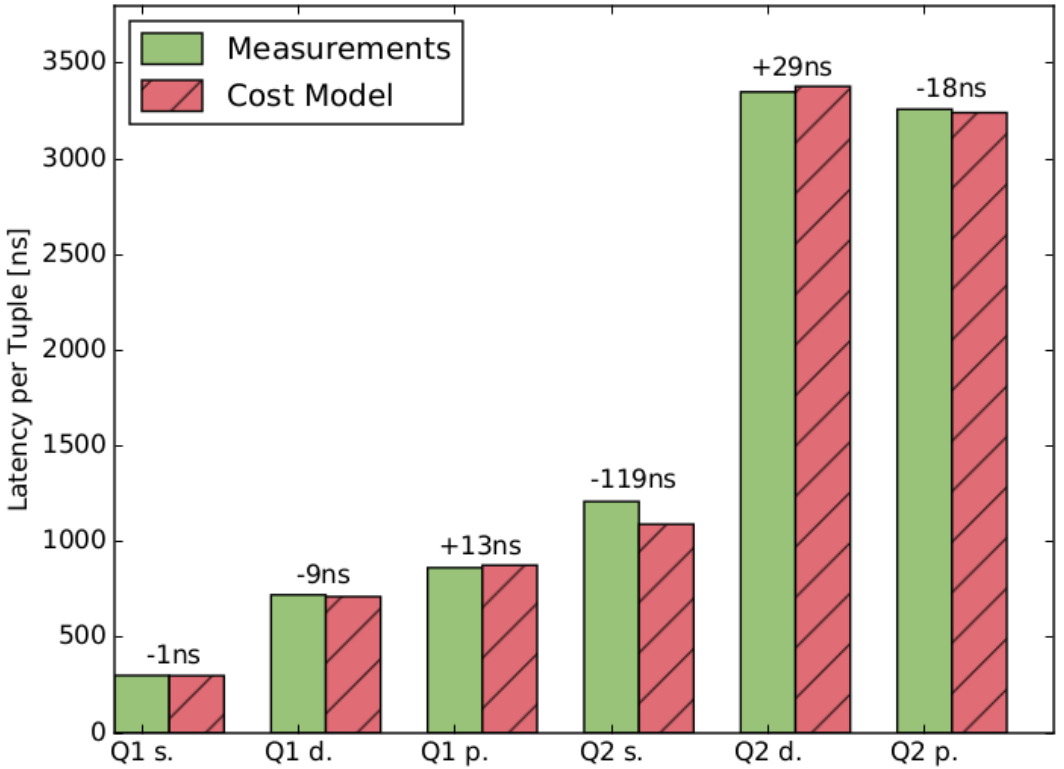
Query Results

Q1, Q2

s = single thread

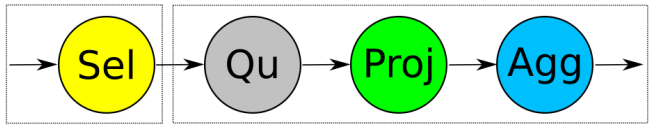
d = decoupled
(Inter-Op.)

p = partitioned
(Intra-Op.)



Example (Q1 d. with Inter-Op. Parallelism)

Inter-Op. Parallelism



Cost formula:

$$c_{Q1} = max\left(\left(c_{op(Sel)} \cdot tp_{\text{in}}\right), \left(c_{op(Proj)} \cdot tp_{\text{in}} + c_{op(Agg)} \cdot tp_{\text{in}}\right)\right) + c_{queue} \cdot tp_{\text{in}}$$

$$c_{Q1} = c_{op(Proj)} \cdot tp_{\text{in}} + c_{op(Agg)} \cdot tp_{\text{in}} + c_{queue} \cdot tp_{\text{in}}$$ // tp_{in} = 20% of 1mio tuples

$$c_{Q1} = \left(c_{op(Proj)} + c_{op(Agg)} + c_{queue}\right) \cdot 200.000$$

$$c_{Q1} = \left(340\,ns + 552\,ns + 2650\,ns\right) \cdot 200.000$$

$$c_{Q1} = 708.400.000\,ns$$

(= Query execution time for processing 1mio tuples)

Operator	Time/Tuple
Projection	340ns
Aggregation	552ns
Queue	2650ns

$Latency_{Q1} = c_{Q1} / 1.000.000$

$= \underline{\sim 709ns / tuple}$

- Accessing Tuple in L1 [~**3**ns],
- Writing result in Main Memory [~**146**ns],
- CPU time (function calls, subscribe/publish tuples,... →clock speed KNL!) [~**191**ns]

Conclusion

- Goal: Cost Model for Query Optimizer
  - Data Stream Processing
  - Modern Hardware (CPU, Memory)

- Cost Model focused first on multithreading aspect
  - Calibration approach for relevant HW factors
  - Parameters of operator implementations (SPE)

- Derived cost model, tested with Xeon Phi KNL

Summary

Cost Model:

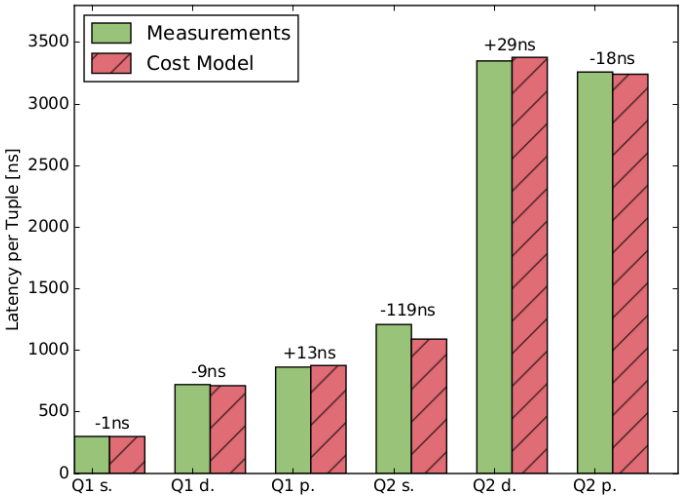
- HW-Calibration approach, measuring relevant HW factors
- Knowledge about SPE: Implementation, Algorithms

(First) focus on multithreading aspect

- **Inter-Operator** & **Intra-Operator** parallelism

Testing:

- *PipeFabric*[1] SPE
- Intel Xeon Phi KNL (7210) processor



[1] https://github.com/dbis-ilm/pipefabric